Exercise 4

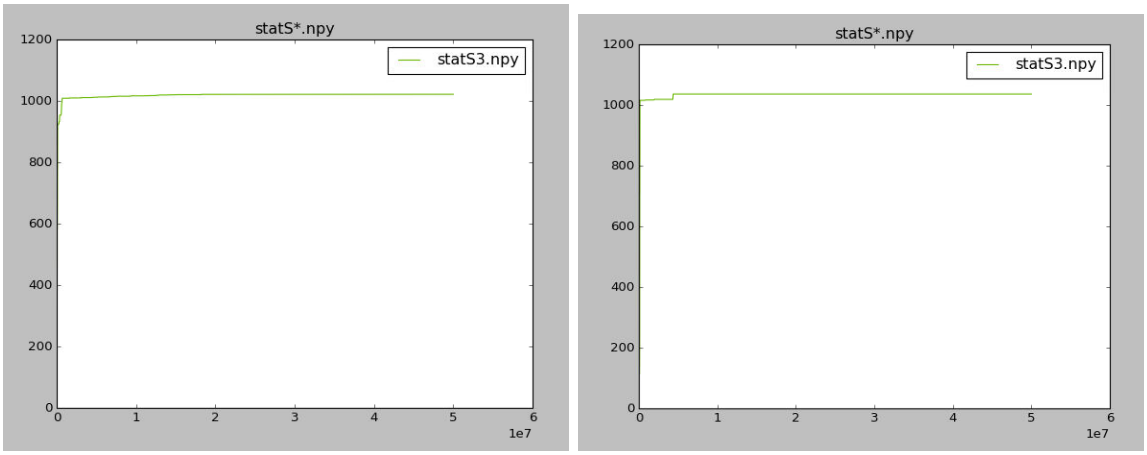
Task description

Evolve the robots with the original reward functions and then compare the behavior of robots evolved with the original and revised reward functions. Could you explain why the original rewards functions are not suitable for evolutionary strategies?

Results

For the exercise the hopper model was used for training and analyzing purposes. A hopper model is presented on the figure 2. The hopper is a two-dimensional one-legged robot that after the jump should balance.

On figure 1, there are the results of training of a hopper model on the original and custom rewards. For both rewards as seed for training it is used Seed = 3.



(a) Custom reward (b) Original reward

Figure 1: The variations of the performance across generations

The average and the standard deviation of performance among multiple runs:

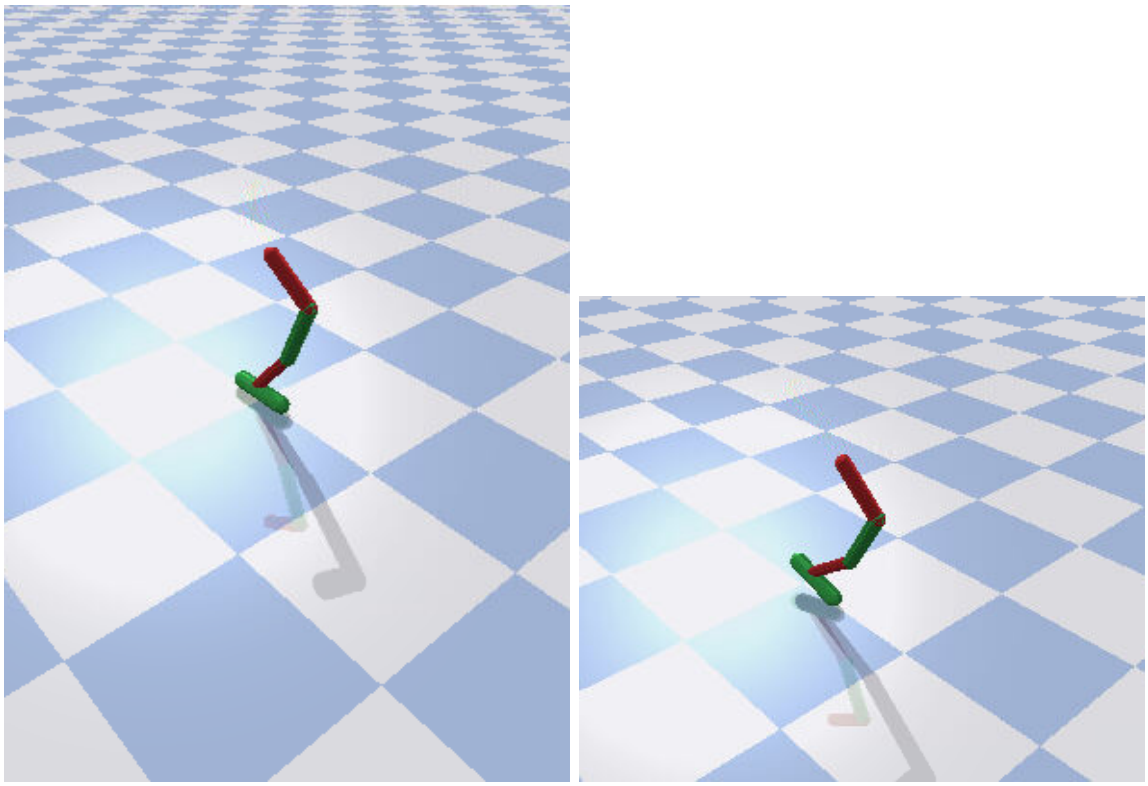
- Custom reward function:

  Average Generalization: 1021.52 +-0.00

- Original reward function:

  Average Generalization: 1036.14 +-0.00

In original version, the reward function estimates through the robot position on the x-axis. On the other hand, a custom version estimates the reward through the potential energy of the



(a) Custom reward function (b) Original reward function

Figure 2: Hopper with different reward calculation after a jump

robot. Through the simulation it can be concluded that he custom reward function wins over the original reward function, because the custom version takes into account the fundamental properties of the dynamical systems motion.

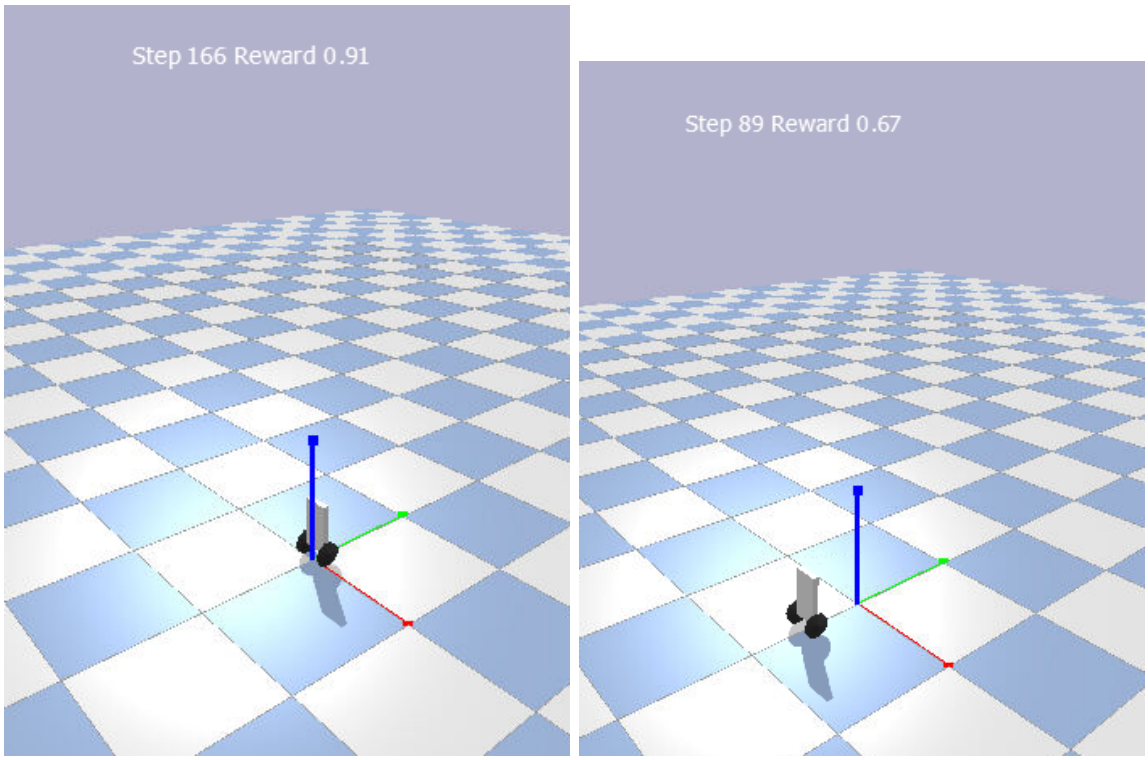
Exercise 5

Task description

Creating and using a new Gym environment from scratch.

Results

The problem considers a wheeled robot called balance bot that can be trained for balance. The balance bot model is presented on the figure 3. The results of the work is presented on the figure 4. The average and the standard deviation of performance among the multiple runs :



(a) Robot simulation 1 (b) Robot simulation 2

Figure 3: Balance bot model

Average Generalization: 169.83 +-0.00

Exercise 6

Task description

Run 10 replications of the experiment from the evorobotpy/xdiscrim folder by using different seeds. To speed-up the training process you can run 2 instances of the program in parallel or eventually more, depending on the characteristics of your computer. Test and analyze the strategy displayed by best robot of each replication. Describe the strategies of the robots by grouping them in families. Try to explain why the robot of each family behave in that manner. Run other experiments by using a feed-forward neural architecture (without memory). Explain how the behavior of evolved robots differ from those evolved with the LSTM architecture (i.e. the Long Short Term Memory architecture).

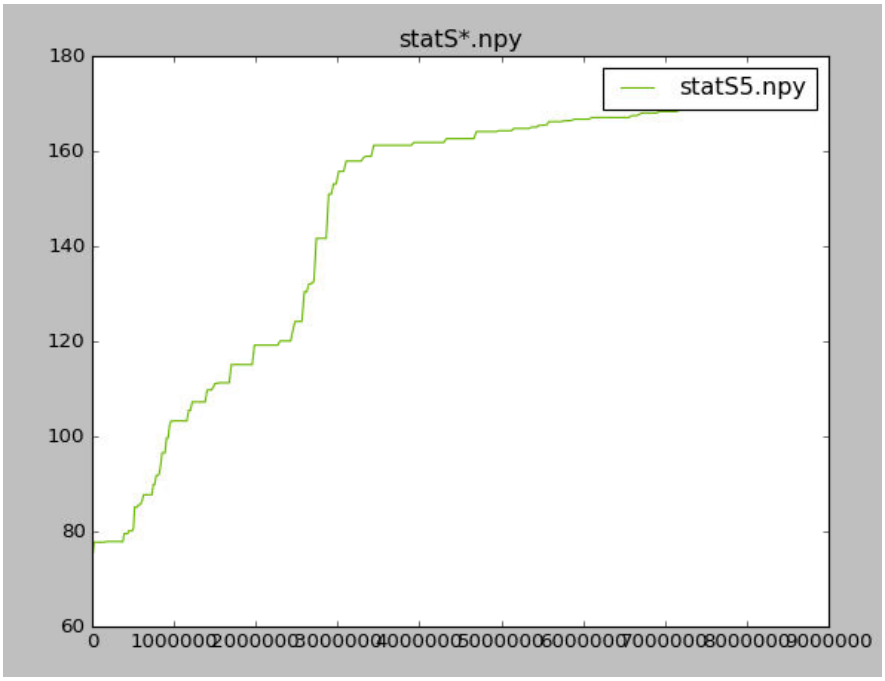


Figure 4: The variations of the performance across generations

Results

On the figure 5 the results of exercise 6 are presented. Seeds from 1 to 10 is for LSTM Neural Network. Seeds from 11 to 13 for feed forward neural network.

The average and the standard deviation of performance among multiple runs:

- LSTM architecture:

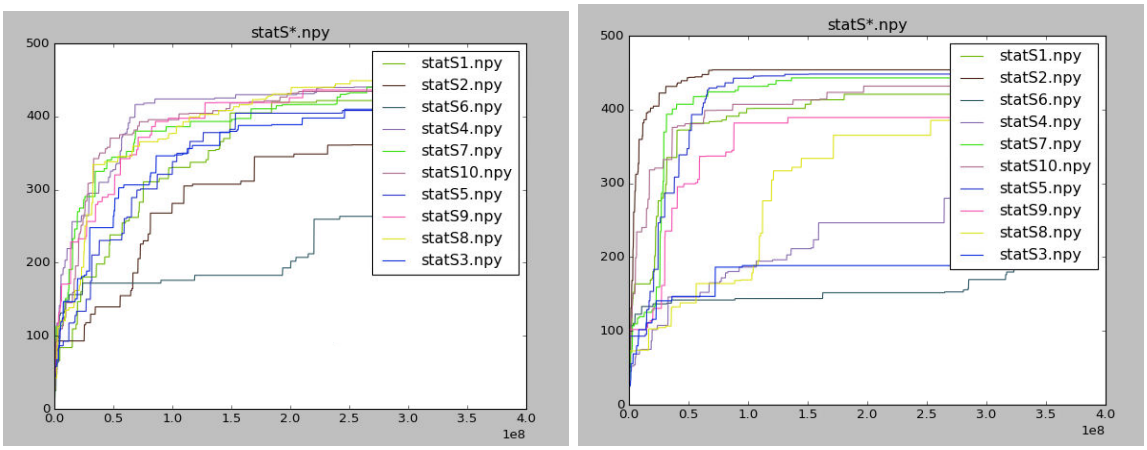
  Average Generalization: 416.28 +-49.89 (10 S*.fit files)

- Feedforward architecture:

  Average Generalization: 366.75 +-96.71 (10 S*.fit files)

To identify the strategies of the robot I run into the problem that the file khepera-wall.sampleSegmentation cannot be opened (core dumped).

Originally, the neural network does not have a memory. Such neural network without memory is feedforward neural network. The weakness of such network is that on each new step it does not take into an account the previous state. That is why the LSTM neural networks were proposed to consider previous state. That is lead to faster and smoother training to reach the target point. That is shown in the figure 5.



(a) LSTM Neural Network (b) Feedforward Neural Network

Figure 5: Balance bot model